

AO-A187 826

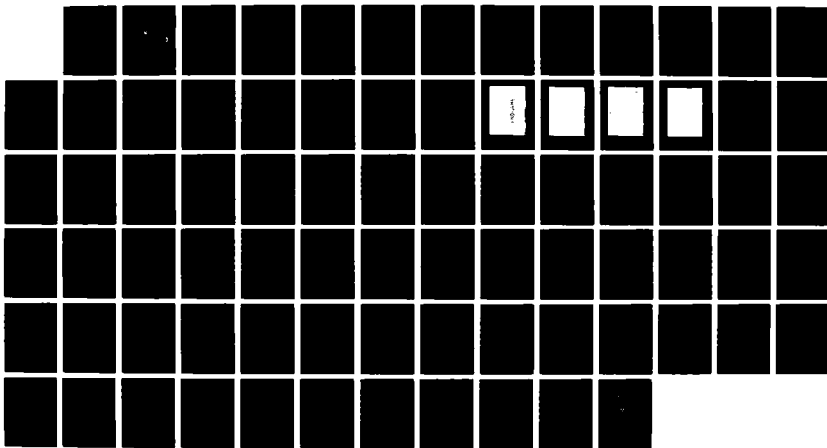
A COMPUTER GRAPHICS SIMULATION OF THE ENDGAME IN  
AIRCRAFT COMBAT SURVIVABILITY(U) NAVAL POSTGRADUATE  
SCHOOL MONTEREY CA Y H LEE DEC 87

1/1

UNCLASSIFIED

F/O 15/6

HL





MICROCOPY RESOLUTION TEST CHART

NS-1963-A

AD-A187 826

②  
DTIC FILE COPY

# NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC  
ELECTE  
JAN 05 1988  
S D E

## THESIS

A COMPUTER GRAPHICS SIMULATION  
OF THE  
ENDGAME IN AIRCRAFT COMBAT SURVIVABILITY

by

Yeong Man Lee

December 1987

Thesis Advisor :

R. E. Ball

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION <b>Unclassified</b>		1b RESTRICTIVE MARKINGS	
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release ; distribution is Unlimited	
2b DECLASSIFICATION/DOWNGRADING SCHEDULE		5 MONITORING ORGANIZATION REPORT NUMBER(S)	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)		7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL (if applicable) Code 67	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	
8a NAME OF FUNDING/SPONSORING ORGANIZATION		9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8b ADDRESS (City, State and ZIP Code)		10 SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO	PROJECT NO
		TASK NO	WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) A COMPUTER GRAPHICS SIMULATION OF THE ENDGAME IN AIRCRAFT COMBAT SURVIVABILITY			
12 PERSONAL AUTHOR(S) Lee, Yeong M.			
13a TYPE OF REPORT Master's Thesis	13b TIME COVERED FROM _____ TO _____	14 DATE OF REPORT (Year Month Day) 1987, December	15 PAGE COUNT 78
16 SUPPLEMENTARY NOTATION			
17 COSATI CODES		18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB GROUP	
		computer graphics simulation program	
		vulnerability, endgame	
19 ABSTRACT (Continue on reverse if necessary and identify by block number)			
<p>A comprehensive computer graphics program was designed for students of the Naval Postgraduate School to assess the vulnerability of air targets to externally detonating missile warheads. This computer simulation program was written in the C language and runs on the IBM PC/AT or compatible. It illustrates the endgame phase of a typical air defense scenario and provides the students the capability to comprehend the dynamic situation.</p>			
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS		21 ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. R. E. Ball		22b TELEPHONE (Include Area Code) (408) 646-2885	22c OFFICE SYMBOL Code 67Bp

Approved for public release; distribution is unlimited.

A Computer Graphics Simulation  
of  
the Endgame in Aircraft Combat Survivability

by

Yeong Man Lee  
Major, Republic Of Korea Air Force  
B.S., Air Force Academy, 1979

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

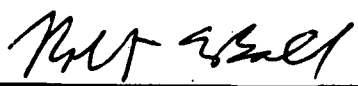
from the

NAVAL POSTGRADUATE SCHOOL  
December 1987


Author:

  
Yeong Man Lee

Approved by:

  
Robert E. Ball, Thesis Advisor

  
M. F. Platzter, Chairman, Department of  
Aeronautics

  
Gordon E. Schacher  
Dean of Science and Engineering

## ABSTRACT

A comprehensive computer graphics program was designed for students of the Naval Postgraduate School to assess the vulnerability of air targets to externally detonating missile warheads. This computer simulation program was written in the C language and runs on the IBM PC/AT or compatible. It illustrates the endgame phase of a typical air defense scenario and provides the students the capability to comprehend the dynamic situation.

Accession For	
NCIS GRA&I	<input checked="checked" type="checkbox"/>
DDIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



## TABLE OF CONTENTS

I.	INTRODUCTION .....	7
II.	THE ENDGAME .....	8
A.	THE MISS DISTANCE .....	8
B.	THE FRAGMENT SPRAY ZONE .....	11
C.	THE BEST POINT FOR DETONATION .....	12
III.	PROGRAMMING "ENDGAME" .....	14
A.	PREPARING SOURCE FILE .....	14
B.	COMPILING .....	14
C.	LINKING .....	15
IV.	GRAPHICS IN THE ENDGAME PROGRAM .....	16
A.	THE CREATED SCREENS .....	17
1.	The Opening Screen .....	17
2.	The Miss Distance Screen .....	18
3.	The Fragment Spray Zone Screen .....	18
4.	The Best Point for Detonation Screen ....	18
B.	THE OPTION SCREENS .....	19
V.	SUMMARY AND RECOMMENDATIONS .....	24
	APPENDIX A. PROGRAM VARIABLES .....	25
	APPENDIX B. PROGRAM LISTINGS .....	27
	APPENDIX C. USER'S MANUAL .....	70
	LIST OF REFERENCES .....	75
	INITIAL DISTRIBUTION LIST .....	76

## LIST OF FIGURES

1.	Encounter Conditions in Groval Coordinates .....	8
2.	The Fragment Spray Zone with Respect to Missile ...	12
3.	The Opening Screen .....	20
4.	The Miss Distance Screen .....	21
5.	The Fragment Spray Zone Screen .....	22
6.	The Best Point for Detonation Screen .....	23



## ACKNOWLEDGEMENTS

I would like to express my deep appreciation to my thesis advisor, Professor Robert E. Ball, for his patient advice and kindful guidance with sharing his knowledgement of this fields. I would also like to thank the Republic of Korea Air Force for giving me the opportunity to study at the Naval Postgraduate School. Finally, I wish to express my thanks to my wife, Cheung Suk, for her assistance and to my son, Ji Woong.

## I. INTRODUCTION

This program is designed to simulate the endgame phase between a missile and an air target for the purpose of illustrating the dynamics of the encounter. This program, called ENDGAME, enables the user to determine the miss distance, the fragment spray zone, and the best point for detonation for a given ordnance package, target and missile velocity and flight path.

This program, written in the C language, and using the HALO graphics subroutine packages, provides the basic framework from which further simulations of increased complexity and sophistication can be easily implemented.

To establish the proper perspective, a brief outline is given as follows; the following chapter gives the theoretical background for the endgame. This is followed in Chapters III and IV by a construction of the program in the C language with comments on graphical features. Finally conclusions are drawn in Chapter V, and recommendations are made for future improvements. The program variables are described in Appendix A. The program listing is given in Appendix B, and a user's manual is given in Appendix C.

## II. THE ENDGAME

### A. THE MISS DISTANCE

For a given encounter between a missile and a target, the guidance system can use any one of several methods to navigate the missile along the flight path from launch to the intercept with the target. The miss distance is basically the minimum distance between the missile and its target during the encounter. It is used for determining the probability that a fragment from the detonation of the warhead hits the target. The miss distance can be evaluated based upon the encounter conditions shown in Figure 2-1 and assuming the velocity of both the missile and the target are constant.

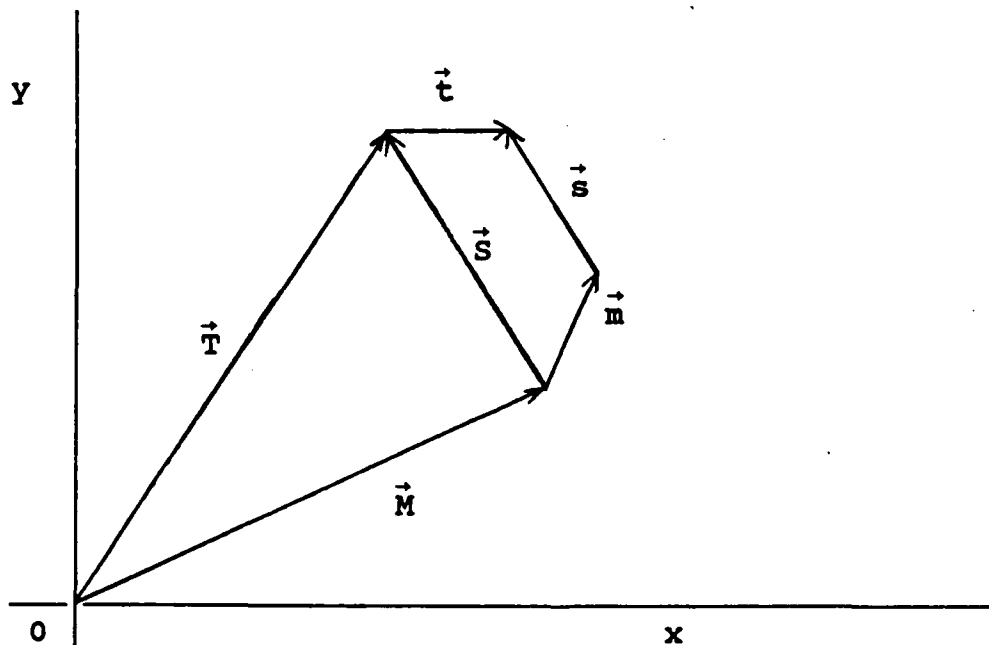


Figure 2-1. Encounter Conditions in Global Coordinates

To evaluate the miss distance, global coordinates are chosen for convenience and proper understanding. See Figure 2-1.

At time  $r$

$$\vec{T} + \vec{t} = \vec{M} + \vec{m} + \vec{s} \quad (1.1)$$

where

$\vec{T}$  = the vector from the origin to the target at  $r=0$

$\vec{t}$  = the vector from the target at  $r=0$  to the target at  $r=r$

$\vec{M}$  = the vector from the origin to the missile at  $r=0$

$\vec{m}$  = the vector from the missile at  $r=0$  to the missile at  $r=r$

$\vec{s}$  = the vector from  $\vec{m}$  to  $\vec{t}$  (the separation vector)

Since the target velocity,  $V_t$ , is assumed to be constant

$$\vec{t} = V_t r \vec{i} \quad (1.2)$$

And since the missile velocity,  $V_m$ , is also assumed to be constant

thus,

$$\vec{m} = V_m (\cos\theta \vec{i} + \sin\theta \vec{j}) r \quad (1.3)$$

$$\begin{aligned} \vec{s} &= (\vec{T} - \vec{M}) + (\vec{t} - \vec{m}) \\ &= [(T_x - M_x) + V_t r - V_m \cos\theta r] \vec{i} \\ &\quad + [(T_y - M_y) - r V_m \sin\theta] \vec{j} \end{aligned} \quad (1.4)$$

where the subscripts  $x$  and  $y$  denotes the  $x$  and  $y$  components

For min  $|\vec{s}|$

$$\frac{d|\vec{s}|}{dr} = 0 \quad (1.5)$$

Solving equation (1.5) for  $r$  using equation (1.4) gives

$$r = \frac{S_x (V_m \cos\theta - V_t) + S_y V_m \sin\theta}{(V_m \cos\theta - V_t)^2 + (V_m \sin\theta)^2} \quad (1.6)$$

Define

$$S_x = T_x - M_x$$

$$S_y = T_y - M_y$$

and

$$V_{mtx} = V_m \cos\theta - V_t$$

$$V_{mty} = V_m \sin\theta$$

Eq (1.6) can be written as

$$r = \frac{S_x V_{mtx} + S_y V_{mty}}{V_{mtx}^2 + V_{mty}^2} \quad (1.7)$$

and eq (1.4) can be given in the form

$$\vec{s} = [S_x - V_{mtx} r] \vec{i} + [S_y - V_{mty} r] \vec{j} \quad (1.8)$$

Equation (1.8) can be used to determine whether the given encounter is an Early Bird or a Late Bird situation. In the Early Bird case, the missile passes in front of the target, whereas it passes behind the target in the Late Bird case.

## B. THE FRAGMENT SPRAY ZONE

When a warhead is detonated near an aircraft, the fragments are usually ejected uniformly and begin to propagate outward in a divergent spherical pattern at a velocity that is the vector sum of the initial fragment velocity from a static warhead detonation and the missile velocity. [Ref.1]

If the fragments are assumed to have uniform velocity, and to be uniformly spread over a spherical segment, the fragment spray density at the distance R is given by

$$\rho = \frac{N}{2\pi R^2 (\cos \phi_1 - \cos \phi_2)} \quad (1.9)$$

where

N = the total number of fragments in the warhead

The equation defining the angles  $\phi_1$  and  $\phi_2$  is

$$\phi_i = \tan^{-1} \left[ \frac{V_m \sin \theta + V_o \sin(\theta + \alpha_i)}{V_m \cos \theta + V_o \cos(\theta + \alpha_i) \pm V_t} \right] - \theta \quad (1.10)$$

$$i = 1, 2$$

where

$V_m$  = the speed of the missile

$\theta$  = the elevation angle of the missile

$V_o$  = the average fragment speed with respect to a stationary warhead

$\alpha_1$  = the static leading fragment spray angle  
 $\alpha_2$  = the static trailing fragment spray angle  
 $V_t$  = the target horizontal speed

### C. THE BEST POINT FOR DETONATION

The fragment spray density from the detonation point is determined by the given detonation distance. At this point, the following problem is encountered. Where is the best point for detonation? The specific answer to that question requires an understanding that there are four different fragment spray zones created by the encounter conditions and the warhead parameters as shown in Figure 2-2.

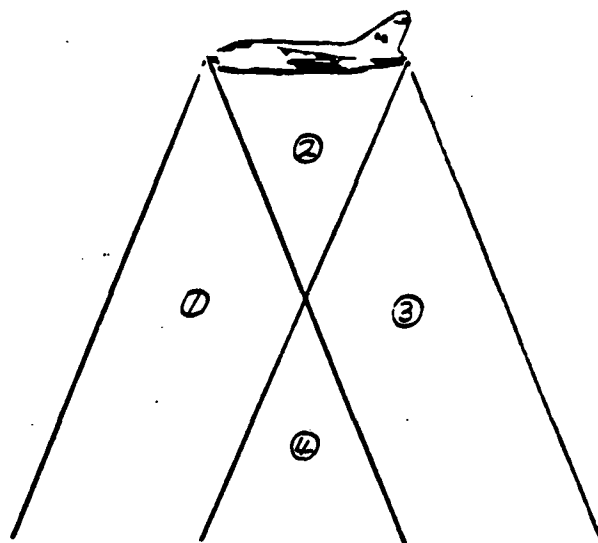


Figure 2-2. The Fragment spray zone

In region 1 and 3, part of the fragment spray zone hits part of the target. In region 2, the entire fragment spray zone hits part of the target, and in region 4, part of the fragment spray zone hits all of the target.



### III. PROGRAMMING "ENDGAME"

This chapter contains the programming description of the implementation of the endgame equations using the Lattice C compiler. Even though several compilers are available these days, the Lattice C compiler is used since it is a portable compiler for the high level programming language called C.

#### A. PREPARING SOURCE FILE

The source file was written by using the text line editor EDLIN in IBM-DOS version 3.2 to create, change, and display the source file. All the external variables are declared outside of the module, including other standard source files. The driver program, `main()`, is the first routine to be executed. This is followed next by several subroutine programs, sub-subroutines, and so on. Since the program is so massive that it is a little hard to read the whole program thoroughly at a glance, each of the subroutine functions are entitled. See the program listing in Appendix B for further particulars.

#### B. COMPILING

The Lattice C compiler version 3.00 [Ref.3] is used for compiling the source file into the object file. The LC

command is used with several options: -w option to shut off warning messages generated for return statements, and -k2 option to specify the generate code for 80286. Another option chosen is -s for naming program segments to provide more availability on RAM working memory. Otherwise the storage requirements exceed the available memory.

This was the most difficult part of the author's thesis work: to find the proper options that would work with the P model as the program got bigger and bigger. Refer to the Lattice C compiler manual for more detailed information.

#### C. LINKING

When linking the object file into the program file, the LINK command in MS-DOS version 3.0 was used. Since most parts of the program call the graphics subroutine functions in HALO to draw the graphics on the screen, a special format for linking the program with the HALO library is required. With program model (version 3.0) supported by the full available working memory, the following command is used:

```
LINK CP+ENDGAME+HALODVP3,ENDGAME,,LCMP+LCP+HALOP3
```

Further details on each step is discussed in Section 4 of the HALO manual and can be referred to by looking into the LINK command in the MS-DOS manual.

#### IV. GRAPHICS IN THE ENDGAME PROGRAM

As mentioned in the Introduction, most of the author's efforts were devoted to creating screens that illustrate the endgame geometry using colorful graphics to obtain desirable results from the given warhead and encounter data input. In ENDGAME, the useful information is presented on each screen with high resolution, is processed and displayed graphically on the screen, simulating each endgame phase and showing what is happening.

All the screens were created using HALO in the graphics mode, except the option screen which shows the sequential questions and input data. The HALO library used in this work, version 2.26, is a collection of high performance subroutines which allow the application programmer to develop sophisticated computer graphics programs.

ENDGAME uses world coordinates throughout the program. The world coordinates are mapped to device coordinates on a specified range from 0 to 640 on x-axis and from 0 to 350 on y-axis so as to locate each point within the resolution of the EGA(Enhanced Graphics Adaptor) display device. The upper left corner of the screen is at coordinates (0,0) and the lower right corner is at coordinates (639,349) for convenience. To draw objects, the coordinates and position of the objects have to be scaled by the aspect

ratio. Aspect is expressed as a ratio of width to height. The value 1.439 was empirically chosen to make the range increment on the horizontal axis be the same as that on the vertical axis.

Another factor considered was trans\_factor [Ref.2]. As mentioned in the user's manual in Appendix C, some particular situations had to be excluded in the endgame. The user should exclude the encounter conditions where the difference in relative position between the target and the missile is larger than 1,000 ft. Hence, the difference always lies within 1,000 ft and the graphic resolution is taken account into by trans\_factor. The value of trans\_factor such as 0.2, 0.3, etc., is empirically chosen based upon the value that gives good graphic resolution on the screen.

#### A. CREATED SCREENS

All the screens created by HALO are in the graphics mode. There are three different screens which allow students to comprehend the dynamic situation between the target and the missile.

##### 1. The Opening Screen

When ENDGAME is executed, the opening screen is displayed, welcoming the student to the endgame assessment as shown in Figure 4-1. Hitting any key allows the student to proceed to the first step.

## 2. The Miss Distance Screen

After inputting the data requested by the option screen, the student will see the miss distance screen (Figure 4-2). This screen tells the student what the miss distance is numerically and shows how it looks vectorically with the target and the missile vectors shown on the screen. It also prints and shows whether the encounter situation is an Early Bird or a Late Bird. Refer to the Appendix A for the other variables shown on the screen.

## 3. The Fragment Spray Zone Screen

This screen (Figure 4-3) shows the fragment spray zone with respect to the target. Unfortunately, the screen doesn't show the different fragment vectors due to leading and trailing fragment spray angle with different colors. However, the reader should be able to visualize the fragment pattern. Also refer to the Appendix A for other variables.

## 4. The Best Point for Detonation Screen

The student will see where the best point for detonation is on this screen (Figure 4-4). It shows the zones which are appropriate for the detonation of the warhead for the given encounter situation considering the combination of the dynamic fragment speed with respect to the target and the missile velocity vector. From this screen, the student can go back to see the previous screens

simply by following the instructions on the bottom of the screen. Finally, the end screen will appear if the student follows the appropriate instruction on the bottom of the screen.

#### B. OPTION SCREENS

Several questions appear on the screen prompting the student to enter the necessary input data. This data is used to calculate the miss distance and the best point for detonation. These screens are not in the graphics mode, but are in standard text mode. Warning or error messages might appear if the student inputs unreasonable data. The limitations for this endgame program are described in the user's manual which is given in Appendix C.

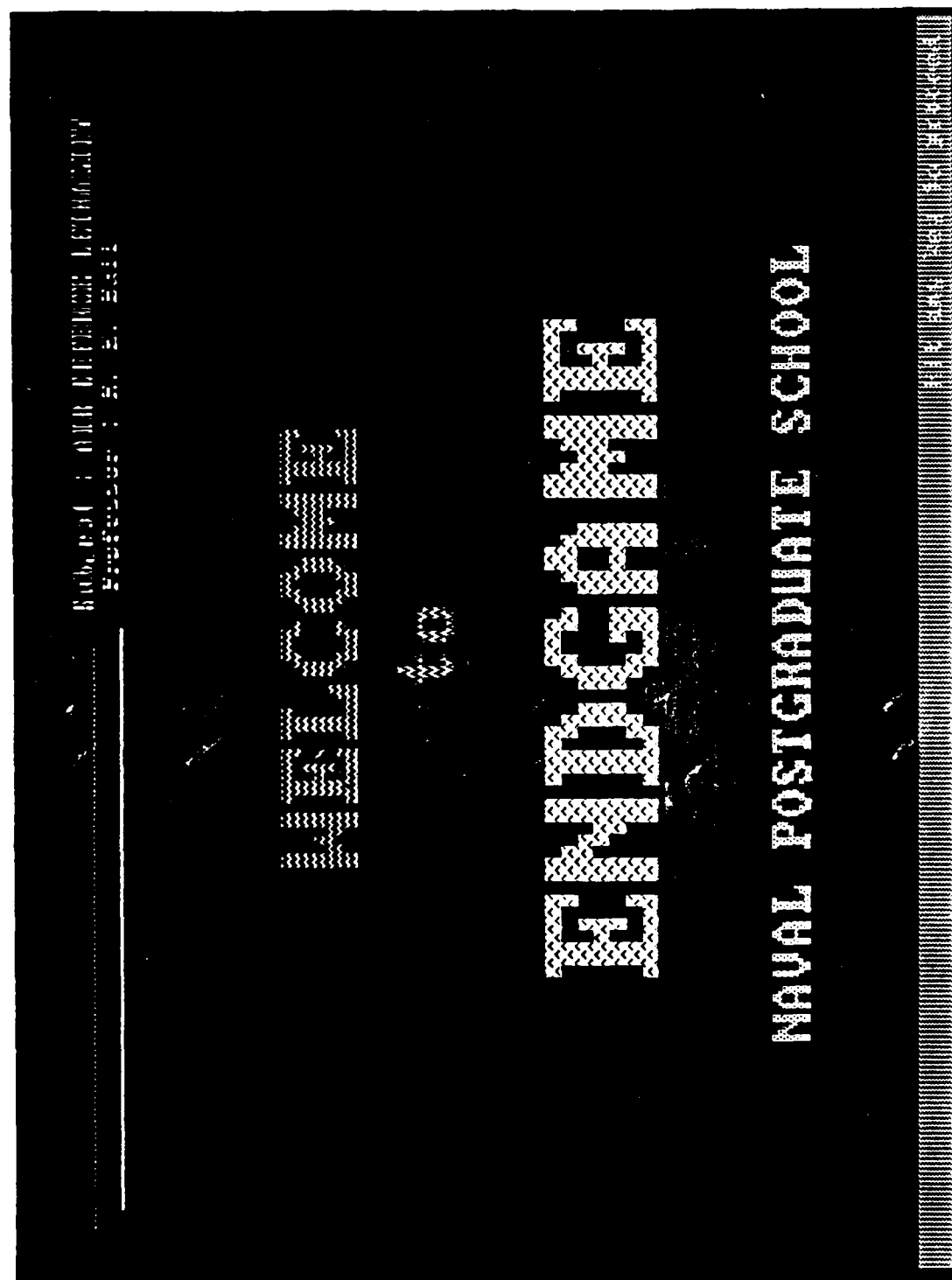


Figure 4-1. The Opening Screen

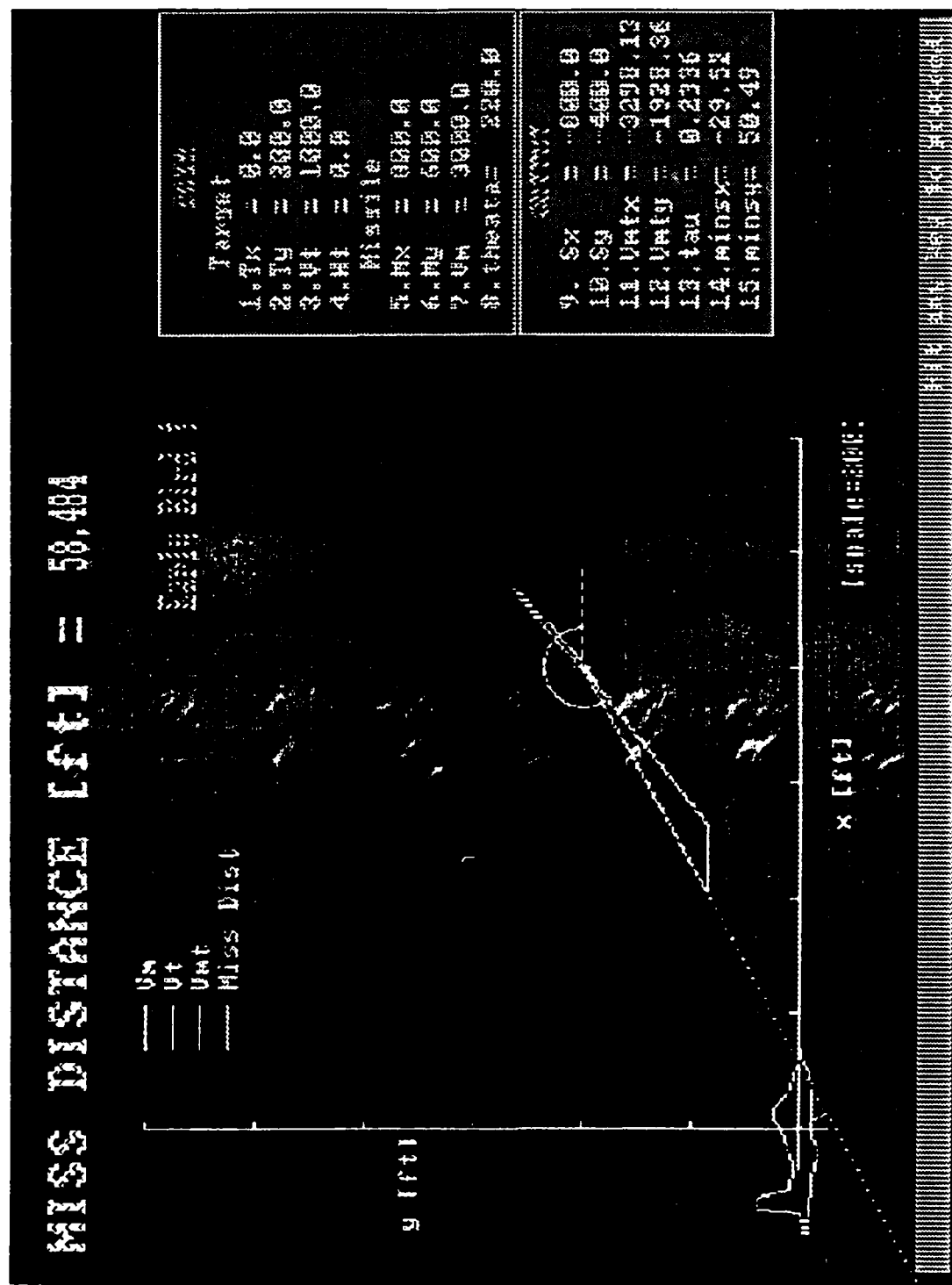


Figure 4-2. The Miss Distance Screen





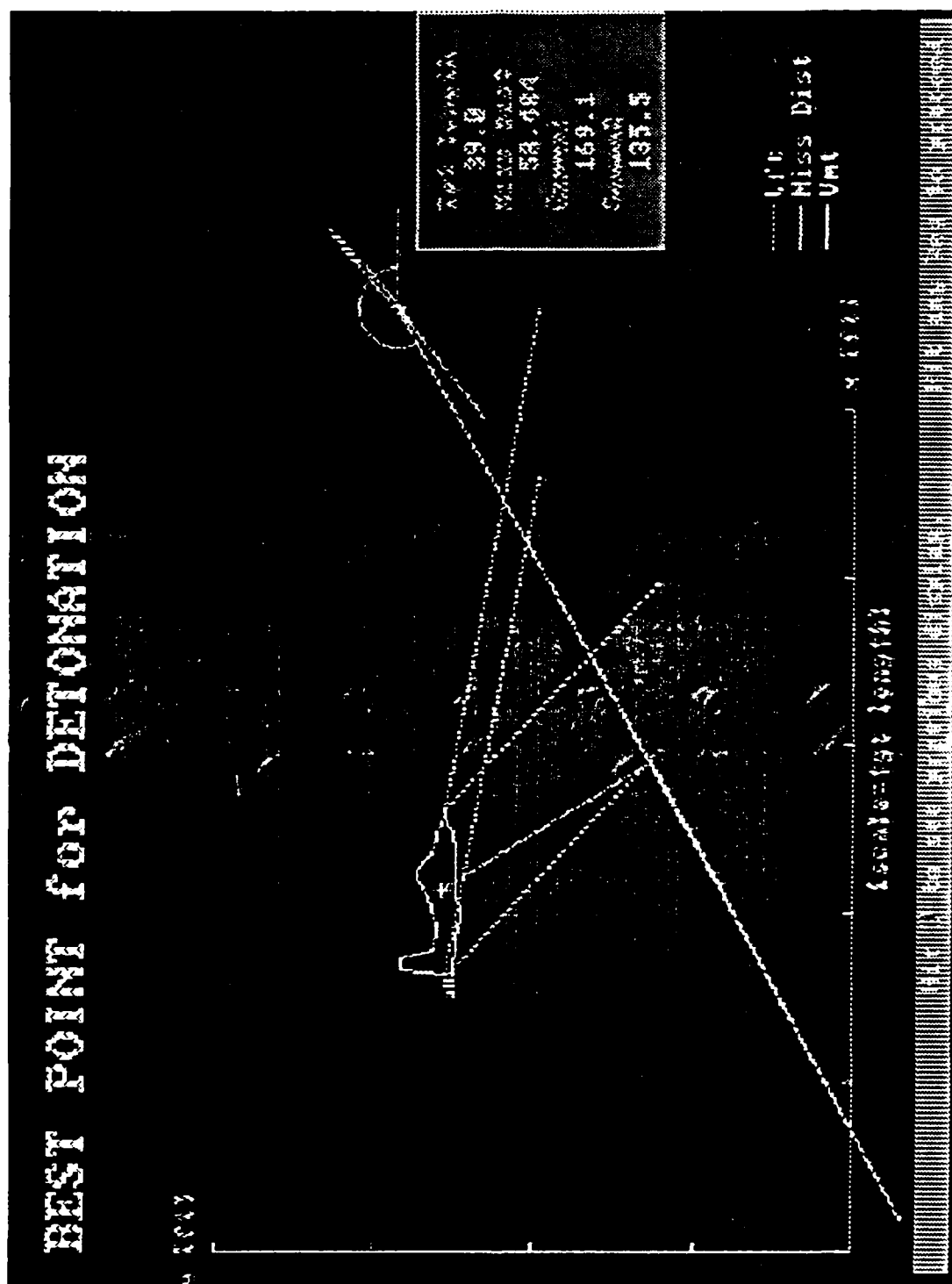


Figure 4-4. The Best Point for Detonation Screen

## V. SUMMARY AND RECOMMENDATIONS

This computer graphics simulation program for the endgame between a missile and an air target was written in the C language using the Lattice C compiler running on MS-DOS. The program is divided into three major categories: miss distance, fragment spray zone, and best point for detonation. It was intended to visually demonstrate the dynamic evaluation for the terminal events in the encounter between an aircraft and a high explosive proximity fuzed warhead on the missile.

It is hoped, finally, that students who are more interested in this field will continue to develop the present ENDGAME program with an anxiety and an eagerness. Especially, a more realistic endeavor to expand the variety of options within the program to account for any encounter situation is strongly recommended. For example, adding a target acceleration to the target motion would allow more realism. From a programming point of view, a solution to the memory expansion problem is also desirable. Hopefully, the probability of kill will be developed in future study.

## APPENDIX A

### PROGRAM VARIABLES

The variables used in endgame computer program are listed as follows;

alpha1 [degree]	static leading fragment spray angle with respect to missile axis
alpha2 [degree]	static trailing fragment spray angle with respect to missile axis
gamma1 [degree]	angle from horizontal to dynamic leading fragment spray
gamma2 [degree]	angle from horizontal to dynamic trailing fragment spray
ht [degree]	target heading
length [ft]	target presented length
minsx [ft]	x component of miss distance vector
minsy [ft]	y component of miss distance vector
miss_distance [ft]	miss distance or closest point
mx [ft]	x component of missile position
my [ft]	y component of missile position
num	total number of fragments in warhead
phone [degree]	dynamic leading fragment spray angle with respect to missile axis
phitwo [degree]	dynamic trailing fragment spray angle with respect to missile axis
r_det [ft]	detonation distance
rho [frag/ft <sup>2</sup> ]	fragment spray density
sx [ft]	x component of difference between missile and target position
sy [ft]	y component of difference between

## missile and target position

tau	[sec]	minimum time for miss distance
theata	[degree]	elevation angle of missile
tx	[ft]	x component of target position
ty	[ft]	y component of target position
vft1	[ft/sec]	dynamic leading fragment speed with respect to target
vft2	[ft/sec]	dynamic trailing fragment speed with respect to target
vftx1	[ft/sec]	x component of vft1
vftx2	[ft/sec]	x component of vft2
vfty1	[ft/sec]	y component of vft1
vfty2	[ft/sec]	y component of vft2
vm	[ft/sec]	speed of missile
vmtx	[ft/sec]	x component of missile speed with respect to target
vnty	[ft/sec]	y component of missile speed with respect to target
vo	[ft/sec]	average fragment speed with respect to stationary warhead
vt	[ft/sec]	target horizontal speed

## APPENDIX B

### PROGRAM LISTINGS

The endgame computer simulating program written in C is as follows.

```

/***** define the all external variables *****/
/***** include another source file *****/

#include "stdio.h"
#include "stdlib.h"
#include "math.h"

/***** image file memorized *****/

static char file1[]={"miss.pic"};
static char file2[]={"prob.pic"};
static char file3[]={"best.pic"};

/***** position variables *****/

float x1=0.0,y1=0.0,x2=640.0,y2=350.0; /*world coordinate*/
float gcx,gcy ; /* graphic cursor position */
float gcx1,gcx2,gcyl,gcyl2; /* for varing graphic cursor */
float tcx,tcy; /* text cursor position */
float dx,dy; /* cursor increment */
float bxl,byl,bx2,by2 ; /* box creation */

/*** variables for adjusting to screen ***/

float trans_factor ;
float dist_factor,dist_1,dist_2;

/***** color definition *****/

int black =0;
int blue = 1 ;
int green = 2 ;
int cyan = 3 ;
int white = 7 ;
int liblue = 9 ;
int ligreen = 10 ;
int licyan = 11 ;
int lired = 12 ;
int lipurple = 13 ;
int liyellow = 14 ;

/***** system variables *****/

float aspect=0.7249; /* ratio of width to height */

```

```

int mode = 4;          /* enhanced graphics adapter mode */
int page=2;           /* available graphics page for system */

/***** drawing variables *****/

float angl,ang2;       /* starting and ending angle of arc */
int width;             /* line width */
int style;             /* line style */
float radius;          /* radius of arc in world coordinates */

char s[15];

int c;
int h,w,p,m;

float itx,ity;
float imx,imy;
int count;

/***** variables for program *****/

float tx,ty ;
float mx,my ;
float sx,sy ;
float vt,vm ;
float vo ;
float ht ;
float theata;
float tau;
float vmtx,vmtx ;
float minsx,minsy ;
float m_dist;
float vftx1,vfty1;
float vftx2,vfty2;
float gammal,gamma2;
float alphas,alpha2;
float vft1,vft2;
float phone,phitwo ;
float r_det,num,length,rho;

```



```

/*****
/**          main function          **/
*****/

/* This states that it is the first or main routine to be
executed. It shows the structure of the endgame program */

main()
{
setdev("haloibme.dev");          /* define graphics system */
setscreen(&page);                 /* set screen type */
initgraphics(&mode);             /* start with HALO */
setasp(&aspect);                 /* set aspect ratio */
setworld(&x1,&y1,&x2,&y2);         /* world coordinates */

opening();                       /* opening screen */
option1();
draw1();                         /* miss distance screen */
option2();
draw2();                         /* fragment spray zone screen */
draw3();                         /* best point for detonation screen */

closegraphics();                /* end up with HALO */
}

```

```

/*****
/**          SUBROUTINE:  OPENING()          **/
*****/

opening()
{
var13();
var14();

setcolor(&lired);
gcx=30.0,gcy=320.0;
movabs(&gcx,&gcy);
gcx=320.0;
lnabs(&gcx,&gcy);

settextclr(&lired,&black);
tcx=340.0,tcy=320.0;
movtcurabs(&tcx,&tcy);
text("Subject : AIR DEFENCE LETHALITY");

setcolor(&lipurple);
gcx=40.0,gcy=310.0;
movabs(&gcx,&gcy);
gcx=330.0;
lnabs(&gcx,&gcy);

settextclr(&lipurple,&black);
tcx=350.0,tcy=310.0;
movtcurabs(&tcx,&tcy);
text("Professor : R. E. Ball");

h=4,w=4;
settext(&h,&w,&p,&m);
settextclr(&green,&black);
tcx=210.0,tcy=220.0;
movtcurabs(&tcx,&tcy);
text("WELCOME");

h=3,w=3;
settext(&h,&w,&p,&m);
settextclr(&ligreen,&black);
tcx=299.0,tcy=185.0;
movtcurabs(&tcx,&tcy);
text("to");

h=6,w=6;
settext(&h,&w,&p,&m);
settextclr(&licyan,&black);
tcx=155.0,tcy=107.0;
movtcurabs(&tcx,&tcy);
text("ENDGAME");

```

```

h=2,w=2;
settext(&h,&w,&p,&m);
settextclr(&liyellow,&black);
tcx=123.0,tcy=55.0;
movtcurabs(&tcx,&tcy);
text("NAVAL POSTGRADUATE SCHOOL");
bottom();
var12();
}

```

```

/*****
/**          SUBROUTINE: OPTION1()          **/
/*****

```

```

option1()
{
lab1:
printf("\n Initial x-position of target [ft] ? :");
scanf("%f",&tx);
if((tx<0.0)|| (tx>100000.0))
{
printf("\n Unreal! Assume that it be above ground and
real.");
printf("\n Type again.");
goto lab1;
}
else
{
lab2:
printf("\n Initial y-position of target [ft] ? :");
scanf("%f",&ty);
if((ty<0.0)|| (ty>100000.0))
{
printf("\n Unreal! Assume that it be above ground and
real.");
printf("\n Type again.");
goto lab2;
}
else
{
lab3:
printf("\n Target speed [ft/sec] ? :");
scanf("%f",&vt);
if((vt<0.0)|| (vt>2000))
{
printf("\n Unreal! Type again.");
goto lab3;
}
else

```

```

{
lab4:
printf("\n Target heading [degree] ? ( type 0 or 180 )
:");
scanf("%f",&ht);
if((ht!=0)&&(ht!=180))
{
printf("\n Assume level flight! Type again.");
goto lab4;
}
else
{
lab5:
printf("\n Initial x-position of missile [ft] ? :");
scanf("%f",&mx);
if((mx<0.0)|| (mx>100000.0))
{
printf("\n Unreal! Assume that it be above ground and
real .");
printf("\n Type again.");
goto lab5;
}
else
{
lab6:
printf("\n Initial y-position of missile [ft] ? :");
scanf("%f",&my);
if((my<0.0)|| (my>100000.0))
{
printf("\n Unreal! Assume that it be above ground and
real.");
printf("\n Type again.");
goto lab6;
}
else if(((abs(tx-mx)<100.0)&&(abs(ty-my)<100.0)) ||
((abs(tx-mx)>1000.0)&&(abs(ty-my)>1000.0)))
{
printf("\n Unproper input data for end game! Start
again.");
goto lab1;
}
else
{
lab7:
printf("\n Missile speed [ft/sec] ? :");
scanf("%f",&vm);
if((vm<vt)|| (vm>6000.0))
{
printf("\n Unreal! Missile speed must be greater than
target speed.");
printf("\n Make it real! Type again.");

```

```

goto lab7;
}
else
{
lab8:
printf("\n Elevation angle of missile (theata) [degree] ?
:");
scanf("%f",&theata);
if (
((ht==0)&&(((tx<mx)&&(ty>my))&&((180<=theata)&&(360>=thea
ta))))
||
(((tx<mx)&&(ty<my))&&((0<=theata)&&(theata<=180))))))
||
((ht==0)&&(((tx>mx)&&(ty>my))&&((theata>=90)&&(theata<=360
))))
||(((tx>mx)&&(ty<my))&&((theata>=0)&&(theata<=270))))))
||
((ht==180)&&(((tx<mx)&&(ty>my))&&((theata>=180)&&(theata<=
90))))
||
(((tx<mx)&&(ty<my))&&((theata>=270)&&(theata<=180))))))
||
((ht==180)&&(((tx>mx)&&(ty>my))&&((theata>=270)&&(theata<=
360))))
||
(((tx>mx)&&(ty<my))&&((theata>=0)&&(theata<=180))))))
))
{
printf("\n Target might not be hit. Put appropriate angle
!!");
goto lab8;
}
else
{
ht = ht * PI / 180.0;
theata = theata * PI / 180.0;
sx = tx - mx ;
sy = ty - my ;
if(ht==0)
vmtx = vm * cos(theata) - vt ;
else
vmtx = vm * cos(theata) + vt ;
vmt y = vm * sin(theata) ;
tau = ( sx * vmtx + sy * vmt y )/(
pow(vmtx,2.0)+pow(vmt y,2.0));
minsx = sx - tau * vmtx ;
minsy = sy - tau * vmt y ;
m_dist = sqrt(pow(minsx,2.0) + pow(minsy,2.0));
ht = ht * 180.0 / PI ;
theata = theata * 180.0 / PI ;

```

```

}
}
}
}
}
}
}
}
}
}

```

```

change:
printf("\n Would you like to change these values ? ( Y/N )
:");
c=getch();
if( c=='y')
{
    printf("\n Yes. Hit any key to change.");
    var12();
    goto lab1;
}
else if( c=='n')
{
    printf("\n Hit any key to continue.");
}
else
{
    printf("\n Type again correctly.");
    goto change;
}
var12();
}

```

```

/*****
/**          SUBROUTINE: DRAW1()          **/
/*****

```

```

draw1()
{
    top();
    text("MISS DISTANCE [ft] =");
    input1();
    input11();
    coord_1();
    loc_1();
    vector1();
    index1();
    output1();
    output11();
    output12();
}

```

```

/*****
/**          SUBROUTINE: OPTION2()          **/
/*****

option2()
{
lab11:
printf("\n Fragment velocity (Vo) [ft/sec] ? :");
scanf("%f",&vo);
if ((vo<0.0)|| (vo>200000.0))
{
printf("\n Unreal!  Assume that it be real.");
printf("\n Type again.");
goto lab11;
}
else
{
lab12:
printf("\n Spray angle (alpha1) [degree] ? :");
scanf("%f",&alpha1);
if ((alpha1<0.0)|| (alpha1>180.0))
{
printf("\n Unreal!  Type again.");
goto lab12;
}
else
{
lab13:
printf("\n Spray angle (alpha2) [degree] ? :");
scanf("%f",&alpha2);
if ((alpha2<0.0)|| (alpha2>180.0))
{
printf("\n Unreal!  Type again.");
goto lab13;
}
else
{
lab14:
printf("\n Number of fragments (N) ? :");
scanf("%f",&num);
if(num<0.0)
{
printf("\n Unreal!  Type again.");
goto lab14;
}
else .
{
lab15:
printf("\n Detonation distance (R) [ft] ? :");
scanf("%f",&r_det);
if (r_det<0.0)

```

```

{
printf("\n Unreal!  Type again.");
goto lab15;
}
else
{
lab16:
printf("\n Target presented length (L) [ft] ? :");
scanf("%f",&length);
if((length<0.0)|| (length>200.0))
{
printf("\n Unreal!  Type again.");
goto lab16;
}
else
{

    alphas1 = alphas1 * PI / 180.0 ;
    alphas2 = alphas2 * PI / 180.0 ;

var15();

    if(ht!=0.0)
        vt=-vt;
    else
        vt=vt;
    vftx1 = vm * cos(theata) + vo * cos(theata + alphas1) -
vt ;
    vftx2 = vm * cos(theata) + vo * cos(theata + alphas2) -
vt ;
    vfty1 = vm * sin(theata) + vo * sin(theata + alphas1) ;
    vfty2 = vm * sin(theata) + vo * sin(theata + alphas2) ;
    gammal = atan(vfty1/vftx1) ;
    gamma2 = atan(vfty2/vftx2) ;
    vft1 = sqrt(pow(vftx1,2.0) + pow(vfty1,2.0));
    vft2 = sqrt(pow(vftx2,2.0) + pow(vfty2,2.0));

var16();

    phione = gammal - theata ;
    phitwo = gamma2 - theata ;

    rho=(num/(2*PI*(pow(r_det,2.0))*(cos(phione)-
cos(phitwo)))));

    }
    }
    }
    }
    }
}

```



```

lab17:
printf("\n Would you like to change these values ?
(Y/N):");
c=getch();
if (c=='y')
{
    printf("\n Yes. Hit any key to continue.");
    var12();
    goto lab11;
}
else if (c=='n');
{
    printf("\n Hit any key to continue.");
}
else
{
    printf("\n Type again correctly.");
    goto lab17;
}
var12();
}

```

```

/*****
/**          SUBROUTINE: DRAW2()          **/
*****/

```

```

draw2()
{
    top();
    text("FRAGMENT SPRAY ZONE");
    input2();
    input21();
    loc_2();
    vector2();
    vector21();
    index2();
    output2();
    output21();
}

```

```

/*****
/**          SUBROUTINE: DRAW3()          **/
*****/

```

```

draw3()
{
    top();
}

```

```

text("BEST POINT for DETONATION");
loc_3();
vector3();
index3();
output3();
output31();
coord_3();
repeat();
}
/*****
/*      SUB-SUBROUTINE  INPUT1()          for draw1()      */
*****/

input1()
{
var14();
setcolor(&white);
bx1=476.0;by1=165.0;bx2=638.0;by2=295.0;
gcx=577.0;gcy=294.0;
boxf();
var13();
settextclr(&ligreen,&blue);
tcx=540.0;tcy=282.0;
movtcurabs(&tcx,&tcy);
text("DATA");
settextclr(&liyellow,&blue);
tcx=510.0;tcy=269.0;
movtcurabs(&tcx,&tcy);
text("Target");
settextclr(&white,&blue);
tcx=490.0;tcy=258.0;
movtcurabs(&tcx,&tcy);
text("1.Tx  =");
tcy=247.0;
movtcurabs(&tcx,&tcy);
text("2.Ty  =");
tcy=236.0;
movtcurabs(&tcx,&tcy);
text("3.Vt  =");
tcy=225.0;
movtcurabs(&tcx,&tcy);
text("4.Ht  =");
settextclr(&liyellow,&blue);
tcx=510.0;tcy=214.0;
movtcurabs(&tcx,&tcy);
text("Missile");
settextclr(&white,&blue);
tcx=490.0;tcy=203.0;
movtcurabs(&tcx,&tcy);
text("5.Mx  =");
tcy=192.0;

```

```

movtcuabs(&tcx,&tcy);
text("6.My  =");
tcy=181.0;
movtcuabs(&tcx,&tcy);
text("7.Vm  =");
tcy=170.0;
movtcuabs(&tcx,&tcy);
text("8.theata=");
}

```

```

/*****
/*      SUB-SUBROUTINE      INPUT11()      for draw1()      */
*****/

```

```

input11()
{
settextclr(&white,&blue);
sprintf(s,"%-8.1f",tx);
tcx=555.0;tcy=258.0;
var10();
sprintf(s,"%-8.1f",ty);
tcy=247.0;
var10();
sprintf(s,"%-6.1f",vt);
tcy=236.0;
var10();
sprintf(s,"%-5.1f",ht);
tcy=225.0;
var10();
sprintf(s,"%-8.1f",mx);
tcy=203.0;
var10();
sprintf(s,"%-8.1f",my);
tcy=192.0;
var10();
sprintf(s,"%-6.1f",vm);
tcy=181.0;
var10();
sprintf(s,"%-5.1f",theata);
tcx=580.0;tcy=170.0;
var10();
ht=ht*PI/180.0;
theata=theata*PI/180.0;
}

```

```

/*****
/*      SUB-SUBROUTINE   COORD_1()      for draw1()      */
/*****

```

```

coord_1()
{
gcx=60.0;gcy=60.0;
movabs(&gcx,&gcy);
gcx=423.5;
lnabs(&gcx,&gcy);
gcx=80.0;gcy=50.0;
movabs(&gcx,&gcy);
gcy=300.0;
lnabs(&gcx,&gcy);

for (count=0;count<6;count++)
{
gcx=gcx+40*1.439;gcy=62.5;
movabs(&gcx,&gcy);
gcy=60.01;
lnabs(&gcx,&gcy);
}
gcy=60.0;
for (count=0;count<6;count++)
{
gcx=80.01;gcy=gcy+40.0;
movabs(&gcx,&gcy);
gcx=84.0;
lnabs(&gcx,&gcy);
}

settextclr(&liyellow);
tcx=230.0;tcy=41.0;
movtcurabs(&tcx,&tcy);
text("x [ft]");
tcx=30.0;tcy=200.0;
movtcurabs(&tcx,&tcy);
text("y [ft]");
settextclr(&lired);
tcx=350.0;tcy=37.0;
movtcurabs(&tcx,&tcy);
text("[scale=200]");
}

```

```

/*****
/*      SUB-SUBROUTINE    LOC_1()          for drawl()      */
*****/

loc_1()
{
var3();
acftl();
msl();
}

/*****
/*      SUB-SUBROUTINE    VECTOR1()        for draw()      */
*****/

vector1()
{
dist_factor=abs(sx)/(cos(atan(abs(sy/sx))));
dist_1=dist_factor*trans_factor/2.5;
dist_2=dist_factor*trans_factor;

/*****      vector for Vm      *****/
setcolor(&licyan);
dx=dist_1*cos(theata);dy=dist_1*sin(theata);
dx=dx*1.439;
lnrel(&dx,&dy);

/*****      vector for Vt      *****/
setcolor(&white);
if ( ht == 0.0 )
    dx=-dist_1*vt/vm;
else
    dx=dist_1*vt/vm;
dy=0.0;
dx=dx*1.439;
lnrel(&dx,&dy);

/*****      vector for Vmt      *****/
inqqcur(&gcx,&gcy,&white);
gcx1=gcx;gcy1=gcy;
setcolor(&liyellow);
gcx=imx+80.0;gcy=imy+60.0;
movabs(&gcx,&gcy);
lnabs(&gcx1,&gcy1);

/*****      extension of Vmt      *****/
if ( ht == 0.0 )
    dx=dist_2*cos(theata)-dist_2*vt/vm;
else

```

```

        dx=dist_2*cos(theata)+dist_2*vt/vm;
        dy=dist_2*sin(theata);
dx=dx*1.439;
style=3;
setlnstyle(&style);
lnrel(&dx,&dy);
style=1;
setlnstyle(&style);
}

```

```

/*****
/*      SUB-SUBROUTINE      INDEX1()      for draw1()      */
*****/

```

```

index1()
{
dx=22.0;dy=0.0;
gcx=120.0;gcy=300.0;
tcx=150.0;tcy=296.0;
in_vm();

gcy=290.0;
tcy=286.0;
in_vt();

gcy=280.0;
tcy=276.0;
in_vmt();

gcy=270.0;
tcy=266.0;
in_miss();
}

```

```

/*****
/*      SUB-SUBROUTINE      OUTPUT1()      for draw1()      */
*****/

```

```

output1()
{
setcolor(&white);
bx1=476.0;by1=70.0;bx2=638.0;by2=163.0;
gcx=577.0;gcy=162.0;
boxf();

settextclr(&ligreen,&blue);

```

```

tcx=535.0;tcy=152.0;
movtcurabs(&tcx,&tcy);
text("OUTPUT");
settextclr(&white,&blue);
tcx=490.0;tcy=141.0;
movtcurabs(&tcx,&tcy);
text("9. Sx   =");
tcy=130.0;
movtcurabs(&tcx,&tcy);
text("10.Sy   =");
tcy=119.0;
movtcurabs(&tcx,&tcy);
text("11.Vmtx =");
tcy=108.0;
movtcurabs(&tcx,&tcy);
text("12.Vmty =");
tcy=97.0;
movtcurabs(&tcx,&tcy);
text("13.tau  =");
tcy=86.0;
movtcurabs(&tcx,&tcy);
text("14.minsx=");
tcy=75.0;
movtcurabs(&tcx,&tcy);
text("15.minsy=");
}

```

```

/*****
/*      SUB-SUBROUTINE  OUTPUT11()      for draw1()      */
/*****

```

```

output11()
{
    sprintf(s,"%-9.1f",sx);
    tcx=572.0;tcy=141.0;
    var10();
    sprintf(s,"%-9.1f",sy);
    tcy=130.0;
    var10();
    sprintf(s,"%-7.2f",vmtx);
    tcy=119.0;
    var10();
    sprintf(s,"%-7.2f",vmty);
    tcy=108.0;
    var10();
    sprintf(s,"%-6.4f",tau);
    tcy=97.0;
    var10();
    sprintf(s,"%-7.2f",minsx);
    tcy=86.0;
}

```

```

var10();
sprintf(s,"%-7.2f",minsy);
tcy=75.0;
var10();
sprintf(s,"%-7.3f",m_dist);
h=2;w=1;
settext(&h,&w,&p,&m);
tcx=360.0;tcy=320.0;
var10();
}

```

```

/*****
/*      SUB-SUBROUTINE      OUTPUT12()      for draw1()      */
/*****

```

```

output12()
{
/*****      miss distance      *****/
setcolor(&liblue);
gcx=itx+80.0;gcy=ity+60.0;
movabs(&gcx,&gcy);
dx=-minsx*1.439*trans_factor;
dy=-minsy*trans_factor;
lnrel(&dx,&dy);

/** determine for early or late bird **/
settextclr(&green,&black);
tcx=340.0;tcy=280.0;
movtcurabs(&tcx,&tcy);
deltcur();
if(((ty>my)&&((ht==0.0)&&(atan(vmtx/vmtx)<atan(sy/sx))))
||((ht>3.12)&&(atan(vmtx/vmtx)>atan(sy/sx))))
||
((ty<my)&&((ht==0.0)&&(atan(vmtx/vmtx)>atan(sy/sx))))
||((ht>3.12)&&(atan(vmtx/vmtx)<atan(sy/sx))))
    text("Early Bird !");
else
    text("Late Bird !");
gwrite(file1);
bottom();
var12();
}

```

```

/*****
/*      SUB-SUBROUTINE      INPUT2()      for draw2()      */
/*****

```

```

input2()

```



```

{
alpha1=alpha1*180.0/PI;
alpha2=alpha2*180.0/PI;

var14();
setcolor(&white);
bx1=460.0;by1=195.0;bx2=638.0;by2=305.0;
gcx=577.0;gcy=299.0;
boxf();

var13();
settextclr(&ligreen,&blue);
tcx=532.0;tcy=291.0;
movtcurabs(&tcx,&tcy);
text("INPUT");
settextclr(&liyellow,&blue);
tcx=495.0;tcy=280.0;
movtcurabs(&tcx,&tcy);
text("Encounter");
settextclr(&white,&blue);
tcx=470.0;tcy=270.0;
movtcurabs(&tcx,&tcy);
text("1.R det  =");
settextclr(&liyellow,&blue);
tcx=495.0;tcy=260.0;
movtcurabs(&tcx,&tcy);
text("Warhead");
settextclr(&white,&blue);
tcx=470.0;tcy=250.0;
movtcurabs(&tcx,&tcy);
text("2.alpha1 =");
tcy=240.0;
movtcurabs(&tcx,&tcy);
text("3.alpha2 =");
tcy=230.0;
movtcurabs(&tcx,&tcy);
text("4.N(frag)=");
tcy=220.0;
movtcurabs(&tcx,&tcy);
text("5. Vo  =");
settextclr(&liyellow,&blue);
tcx=495.0;tcy=210.0;
movtcurabs(&tcx,&tcy);
text("Target");
settextclr(&white,&blue);
tcx=470.0;tcy=200.0;
movtcurabs(&tcx,&tcy);
text("6.length =");
}

```

```

/*****
/*      SUB-SUBROUTINE      INPUT21()      for draw2()      */
/*****

```

```

input21()
{
  sprintf(s,"%-5.1f",r_det);
  tcx=555.0;tcy=270.0;
  var10();
  sprintf(s,"%-6.1f",alpha1);
  tcy=250.0;
  var10();
  sprintf(s,"%-6.1f",alpha2);
  tcy=240.0;
  var10();
  sprintf(s,"%-7.1f",num);
  tcy=230.0;
  var10();
  sprintf(s,"%-7.1f",vo);
  tcy=220.0;
  var10();
  sprintf(s,"%-5.1f",length);
  tcy=200.0;
  var10();
}

```

```

/*****
/*      SUB-SUBROUTINE      LOC_2()      for draw2()      */
/*****

```

```

loc_2()
{
  var7();
  msl();
}

```

```

/*****
/*      SUB-SUBROUTINE      VECTOR2()      for draw2()      */
/*****

```

```

vector2()
{
  alpha1=alpha1*PI/180.0;
  alpha2=alpha2*PI/180.0;
}

```

```
gcy=imy+80.0;gcy=imy+60.0;
movabs(&gcy,&gcy);
```

```
/****** vector for Vo due to alphas *****/
setcolor(&ligreen);
dx=dist_1*(vo/vm)*cos(theata+alphal);
dy=dist_1*(vo/vm)*sin(theata+alphal);
dx=dx*1.439;
lnrel(&dx,&dy);
```

```
/****** vector for Vm *****/
setcolor(&licyan);
dx=dist_1*cos(theata);
dy=dist_1*sin(theata);
var9();
```

```
/****** vector for Vi *****/
setcolor(&lipurple);
dx=dist_1*1.439*(cos(theata)+(vo/vm)*cos(theata+alphal));
dy=dist_1*(sin(theata)+(vo/vm)*sin(theata+alphal));
lnrel(&dx,&dy);
```

```
/****** vector for Vt *****/
setcolor(&white);
dx=-dist_1*vt/vm;
dy=0.0;
dx=dx*1.439;
lnrel(&dx,&dy);
```

```
/****** vector for Vft *****/
setcolor(&lired);
inqgcur(&gcy,&gcy,&lired);
gcy1=gcy;gcy1=gcy;
gcy=imy+80.0;gcy=imy+60.0;
lnabs(&gcy,&gcy);
}
```

```
/******
/* SUB-SUBROUTINE VECTOR21() for draw2() */
/******
```

```
vector21()
{
/****** vector for Vo due to alphas *****/
setcolor(&ligreen);
dx=dist_1*(vo/vm)*cos(theata+alpha2);
dy=dist_1*(vo/vm)*sin(theata+alpha2);
```

```

dx=dx*1.439;
lnrel(&dx,&dy);

/*****          vector for Vm          *****/
setcolor(&licyan);
dx=dist_1*cos(theata);
dy=dist_1*sin(theata);
var9();

/*****          vector for Vi          *****/
setcolor(&lipurple);
dx=dist_1*1.439*(cos(theata)+(vo/vm)*cos(theata+alpha2));
dy=dist_1*(sin(theata)+(vo/vm)*sin(theata+alpha2));
lnrel(&dx,&dy);

/*****          vector for Vt          *****/
setcolor(&white);
dx=-dist_1*vt/vm;
dy=0.0;
dx=dx*1.439;
lnrel(&dx,&dy);

/*****          vector for vft          *****/
setcolor(&lired);
ingcur(&gcx,&gcy,&lired);
gcx2=gcx;gcy2=gcy;
gcx=imx+80.0;gcy=imy+60.0;
lnabs(&gcx,&gcy);

/****          triangle for fragment spray zone          *****/
movabs(&gcx2,&gcy2);
lnabs(&gcx1,&gcy1);
settextclr(&lired,&black);
tcx=15.0;tcy=37.0;
movtcurabs(&tcx,&tcy);
text("Triangle with red lines shows fragment spray zone");
tcy=25.0;
movtcurabs(&tcx,&tcy);
text("with respect to target!");
}

/*****
/*      SUB-SUBROUTINE      INDEX2()      for draw2()      */
*****/

index2()
{
dx=22.0;dy=0.0;
gcx=20.0;gcy=300.0;
tcx=47.0;tcy=296.0;

```

```

in_vo();

gcy=290.0;
tcy=286.0;
in_vm();

gcy=280.0;tcy=276.0;
in_vi();

gcy=270.0;tcy=266.0;
in_vt();

gcy=260.0;tcy=256.0;
in_vft();
}

```

```

/*****
/*      SUB-SUBROUTINE      OUTPUT2()      for draw2()      */
/*****

```

```

output2()
{
setcolor(&white);
bx1=460.0;by1=47.0;bx2=638.0;by2=173.0;
gcx=577.0;gcy=172.0;
boxf();

```

```

var13();
settextclr(&ligreen,&blue);
tcx=530.0;tcy=160.0;
movtcurabs(&tcx,&tcy);
text("OUTPUT");
settextclr(&white,&blue);
tcx=470.0;tcy=150.0;
movtcurabs(&tcx,&tcy);
text("1.Vftx1  =");
tcy=140.0;
movtcurabs(&tcx,&tcy);
text("2.Vfty1  =");
tcy=130.0;
movtcurabs(&tcx,&tcy);
text("3.Vftx2  =");
tcy=120.0;
movtcurabs(&tcx,&tcy);
text("4.Vfty2  =");
tcy=110.0;
movtcurabs(&tcx,&tcy);
text("5.Vft1   =");
tcy=100.0;
movtcurabs(&tcx,&tcy);

```

```

text("6.Vft2   =");
tcy=90.0;
movtcurabs(&tcx,&tcy);
text("7.Gammal =");
tcy=80.0;
movtcurabs(&tcx,&tcy);
text("8.Gamma2 =");
tcy=70.0;
movtcurabs(&tcx,&tcy);
text("9.Phone =");
tcy=60.0;
movtcurabs(&tcx,&tcy);
text("10.Phitwo=");
tcy=50.0;
movtcurabs(&tcx,&tcy);
text("11.Rho   =");
}

```

```

/*****
/*      SUB-SUBROUTINE      OUTPUT21()      for draw2()      */
/*****

```

```

output21()
{
gammal=gammal*180.0/PI;
gamma2=gamma2*180.0/PI;
phone=phone*180.0/PI;
phitwo=phitwo*180.0/PI;

sprintf(s,"%-8.2f",vftx1);
tcx=556.0;tcy=150.0;
var10();
sprintf(s,"%-8.2f",vfty1);
tcy=140.0;
var10();
sprintf(s,"%-8.2f",vftx2);
tcy=130.0;
var10();
sprintf(s,"%-8.2f",vfty2);
tcy=120.0;
var10();
sprintf(s,"%-8.2f",vft1);
tcy=110.0;
var10();
sprintf(s,"%-8.2f",vft2);
tcy=100.0;
var10();
sprintf(s,"%-6.1f",gammal);

```

```

tcy=90.0;
var10();
sprintf(s,"%-6.1f",gamma2);
tcy=80.0;
var10();
sprintf(s,"%-6.1f",phione);
tcy=70.0;
var10();
sprintf(s,"%-6.1f",phitwo);
tcy=60.0;
var10();
sprintf(s,"%-7.4f",rho);
tcy=50.0;
var10();

```

```

gwrite(file2);
bottom();
var12();
}

```

```

/*****
/*      SUB-SUBROUTINE      LOC_3()      for draw3()      */
/*****

```

```

loc_3()
{
trans_factor=0.02*1.2;
var8();
acft1();
}

```

```

/*****
/*      SUB-SUBROUTINE      VECTOR3()      for draw3()      */
/*****

```

```

vector3()
{
/****   vector for Vft from tail section of aircraft   ****/
dx=0.0;dy=0.0;
movrel(&dx,&dy);
dx=-trans_factor*vftx1*1.439;
dy=-trans_factor*vftyl;
lnrel(&dx,&dy);
dx=-dx;dy=-dy;
movrel(&dx,&dy);
dx=-trans_factor*vftx2*1.439;

```

```

dy=-trans_factor*vfty2;
lnrel(&dx,&dy);
dx=-dx;dy=-dy;
movrel(&dx,&dy);

/**** vector for Vft from pitot tube of aircraft ****/
dx=84.0;dy=0.0;
if (ht==0.0)
    dx=dx;
else
    dx=-dx;
movrel(&dx,&dy);
dx=-trans_factor*vftx1*1.439;
dy=-trans_factor*vfty1;
lnrel(&dx,&dy);
dx=-dx;dy=-dy;
movrel(&dx,&dy);
dx=-trans_factor*vftx2*1.439;
dy=-trans_factor*vfty2;
lnrel(&dx,&dy);
mdist();
vmt_vector();
msl_pt();
}

```

```

/*****
/* SUB-SUBROUTINE INDEX3() for draw3() */
*****/

index3()
{
varl3();
dx=22.0;dy=0.0;
gcy=520.0;gcy=70.0;
tcx=547.0;tcy=66.0;
in_vft();

gcy=60.0;tcy=56.0;
in_miss();

gcy=50.0;tcy=46.0;
in_vmt();
}

```

```

/*****
/* SUB-SUBROUTINE OUTPUT3() for draw3() */
*****/

```



```

output3()
{
setcolor(&white);
bx1=520.0;by1=110.0;bx2=639.0;by2=200.0;
gcy=577.0;gcy=199.0;
boxf();

settextclr(&lgreen,&blue);
tcx=540.0;tcy=185.0;
movtcurabs(&tcx,&tcy);
text("Tgt length");

tcy=165.0;
movtcurabs(&tcx,&tcy);
text("Miss Dist");

tcy=145.0;
movtcurabs(&tcx,&tcy);
text("Gammal");

tcy=125.0;
movtcurabs(&tcx,&tcy);
text("Gamma2");
}

```

```

/*****
/*      SUB-SUBROUTINE      OUTPUT31()      for draw3()      */
*****/

```

```

output31()
{
settextclr(&white,&blue);
sprintf(s,"%-5.1f",length);
tcx=560.0;tcy=175.0;
var10();
sprintf(s,"%-7.3f",m_dist);
tcy=155.0;
var10();
sprintf(s,"%-6.1f",gammal);
tcy=135.0;
var10();
sprintf(s,"%-6.1f",gamma2);
tcy=115.0;
var10();
}

```

```

/*****
/*      SUB-SUBROUTINE      COORD_3()      for draw3()      */
/*****

```

```

coord_3()
{
  gcx=20.0;gcy=40.0;
  movabs(&gcx,&gcy);
  gcx=440.0;
  lnabs(&gcx,&gcy);
  gcx=20.0;gcy=40.0;
  movabs(&gcx,&gcy);
  gcy=272.0;
  lnabs(&gcx,&gcy);

  for (count=0;count<5;count++)
  {
    gcx=gcx+84.0;gcy=42.5;
    movabs(&gcx,&gcy);
    gcy=40.01;
    lnabs(&gcx,&gcy);
  }
  gcy=40.0;
  for (count=0;count<4;count++)
  {
    gcx=20.01;gcy=gcy+84/1.439;
    movabs(&gcx,&gcy);
    gcx=24.0;
    lnabs(&gcx,&gcy);
  }

```

```

  settxtclr(&ligreen);
  tcx=450.0;tcy=38.0;
  movtcurabs(&tcx,&tcy);
  text("x [ft]");
  tcx=3.0;tcy=280.0;
  movtcurabs(&tcx,&tcy);
  text("y [ft]");
  tcx=200.0;tcy=27.0;
  movtcurabs(&tcx,&tcy);
  text("[scale=tgt length]");
  deltcx();
}

```

```

/*****
/*      SUB-SUBROUTINE      REPEAT()      for draw3()      */
/*****

```

```

repeat()

```

```

{
gwrite(file3);
iter_1:
bottom();
bottom1();
iter_3:
c=getch();
if (c=='f')
{
iter_2:
setcolor(&black);
clr();
gread(file2);
bottom();
bottom1();
iter_4:
c=getch();
if (c=='f')
{
setcolor(&black);
clr();
gread(file1);
bottom();
c=getch();
goto iter_2;
}
else
{
setcolor(&black);
clr();
gread(file3);
goto iter_1;
}
}
else
{
setcolor(&black);
clr();
close();
}
}

```

```

/*****
/*****      MISCELLANEOUS SUBROUTINE FUNCTION      *****/
/*****

```

```

/*****      aircraft function      *****/

```

```

acft1()
{
var14();
setcolor(&white);
gcx=itx+80.0;gcy=ity+60.0;
movabs(&gcx,&gcy);
setcolor(&liyellow);

dx=3.0;dy=0.0;
var2();
dx=-6.0;dy=0.0;
var2();
dx=3.0;dy=0.0;
var2();
dx=0.0;dy=-3.0;
var2();
dx=0.0;dy=6.0;
var2();
dx=0.0;dy=-3.0;
var2();
setcolor(&white);
if (ht==0.0)
{
    gcx=itx+36.0;gcy=ity+56.0;
}
else
{
    gcx=itx+124.0;gcy=ity+56.0;
}

movabs(&gcx,&gcy);
dx=0.0;dy=3.0;
var2();
dx=2.0;dy=0.0;
var2();
dx=2.0;dy=16.0;
var2();
dx=6.0;dy=0.0;
var2();
dx=4.0;dy=-12.0;
var2();
dx=12.0;dy=-1.0;
var2();
dx=14.0;dy=3.0;
var2();
dx=10.0;dy=4.0;
var2();
dx=4.0;dy=0.0;
var2();
dx=10.0;dy=-6.0;
var2();

```

```

dx=8.0;dy=-1.5;
var2();
dx=8.0;dy=-2.5;
var2();
dx=4.0;dy=0.0;
var2();
dx=0.0;dy=-0.5;
var2();
dx=-4.0;dy=0.0;
var2();
dx=-8.0;dy=-2.5;
var2();
dx=-8.0;dy=-1.0;
var2();
dx=-24.0;dy=0.0;
var2();
dx=-4.0;dy=-2.0;
var2();
dx=-6.0;dy=0.0;
var2();
dx=-10.0;dy=2.0;
var2();
dx=-18.0;dy=1.0;
var2();
dx=-2.0;dy=0.0;
var2();
setcolor(&lired);
dx=-9.0;dy=0.25;
var2();
dx=9.0;
var2();
dx=-7.5;
var2();
dx=7.5;
var2();
dx=-8.4;
var2();
dx=8.4;
var2();
dx=-9.1;
var2();
dx=9.1;
var2();
dx=-9.0;
var2();
dx=9.0;
var2();
}

```

```

/*****
msl()

```

```

missile function

```

```

*****/

```

```

{
var14();
setcolor(&white);
gcx=imx+80.0;gcy=imy+60.0;
movabs(&gcx,&gcy);
msl_point();
gcx=imx+80.0;gcy=imy+60.0;
movabs(&gcx,&gcy);
msl_in_1_point();
}

```

```

/***** missile point function *****/

```

```

msl_point()
{
dx=10.0;dy=0.0;
var6();
dx=-10.0;dy=2.0;
var1();
dx=-30.0;dy=0.0;
var1();
dx=0.0;dy=-2.0;
var1();
dx=40.0;dy=0.0;
var6();
dx=-10.0;dy=-2.0;
var1();
dx=-30.0;dy=0.0;
var1();
dx=0.0;dy=2.0;
var1();

```

```

setcolor(&lired);
dx=-20.0;dy=0.0;
var1();
dx=20.0;dy=0.5;
var6();
dx=-18.0;dy=0.0;
var1();
dx=18.0;dy=-1.0;
var6();
dx=-18.0;dy=0.0;
var1();
dx=18.0;dy=1.5;
var6();
dx=-16.0;dy=0.0;
var1();
dx=16.0;dy=-2.0;
var6();
dx=-16.0;dy=0.0;
var1();
dx=16.0;dy=2.5;

```

```

var6();
dx=-14.0;dy=0.0;
var1();
dx=14.0;dy=-3.0;
var6();
dx=-14.0;dy=0.0;
var1();
}

```

```

/***** missile initial point function *****/
msl_in_1_point()

```

```

{
setcolor(&cyan);
dx=-3.0;dy=0.0;
lnrel(&dx,&dy);
dx=6.0;dy=0.0;
lnrel(&dx,&dy);
dx=-3.0;dy=0.0;
lnrel(&dx,&dy);
dx=0.0;dy=-3.0;
lnrel(&dx,&dy);
dx=0.0;dy=6.0;
lnrel(&dx,&dy);
dx=0.0;dy=-3.0;
lnrel(&dx,&dy);
dx=50.0;dy=0.0;
lnrel(&dx,&dy);
dx=-50.0;dy=0.0;
lnrel(&dx,&dy);

inqqcur(&gcx,&gcy,&cyan);
gcxl=gcx;gcyl=gcy;

dx=50.0*cos(theata)*1.439;
dy=50.0*sin(theata);
lnrel(&dx,&dy);
movabs(&gcxl,&gcyl);

radius=20.0;
angl=0.0;ang2=theata;
arc(&radius,&angl,&ang2);
movabs(&gcxl,&gcyl);
}

```

```

/***** miss distance vector function *****/
mdist()

```

```

{
setcolor(&liblue);
gcx=itx+80.0;gcy=ity+60.0;

```

```

movabs(&gcy, &gcy);
dx=-84.0*minsx/length;
dy=-(84.0*minsy/length)/1.439;
lnrel(&dx, &dy);
}

```

```

/***** missile velocity vector function *****/
vmt_vector()
{
setcolor(&liyellow);
dx=vmtx*2*trans_factor*1.439;
dy=vmtx*2*trans_factor;
lnrel(&dx, &dy);

dx=-2.0*dx;
dy=-2.0*dy;
lnrel(&dx, &dy);
}

```

```

/***** variable missile point function *****/
msl_pt()
{
varl4();
setcolor(&white);
ingcur(&gcy, &gcy, &white);
gcy1=gcy; gcy1=gcy;
msl_point();
movabs(&gcy1, &gcy1);
msl_in_1_point();
}

```

```

/***** index of Vft *****/
in_vft()
{
setcolor(&lired);
movabs(&gcy, &gcy);
lnrel(&dx, &dy);
settextclr(&lired, &black);
movtcurabs(&tcx, &tcy);
text("Vft");
}

```

```

/***** index of vmt *****/
in_vmt()
{
setcolor(&liyellow);
movabs(&gcy, &gcy);

```



```

lnrel(&dx,&dy);
settextclr(&liyellow,&black);
movtcurabs(&tcx,&tcy);
text("Vmt");
}

```

```

/*****          index of miss distance          *****/
in_miss()
{
setcolor(&liblue);
movabs(&gcx,&gcy);
lnrel(&dx,&dy);
settextclr(&liblue,&black);
movtcurabs(&tcx,&tcy);
text("Miss Dist");
}

```

```

/*****          index of vi          *****/
in_vi()
{
setcolor(&lipurple);
movabs(&gcx,&gcy);
lnrel(&dx,&dy);
settextclr(&lipurple,&black);
movtcurabs(&tcx,&tcy);
text("Vi");
}

```

```

/*****          index of missile velocity          *****/
in_vm()
{
setcolor(&licyan);
movabs(&gcx,&gcy);
lnrel(&dx,&dy);
settextclr(&licyan,&black);
movtcurabs(&tcx,&tcy);
text("Vm");
}

```

```

/*****          index of target velocity          *****/
in_vt()
{
setcolor(&white);
movabs(&gcx,&gcy);
lnrel(&dx,&dy);
settextclr(&white,&black);
movtcurabs(&tcx,&tcy);
}

```

```
text("Vt");
}
```

```

/*****          index of fragment velocity          *****/
in_vo()
{
setcolor(&ligreen);
movabs(&gcx,&gcy);
lnrel(&dx,&dy);
settextclr(&ligreen,&black);
movtcurabs(&tcx,&tcy);
text("Vo");
}

```

```

/*****          top marking          *****/
top()
{
h=2;w=2;
settext(&h,&w,&p,&m);
settextclr(&licyan,&black);
tcx=20.0;
tcy=320.0;
movtcurabs(&tcx,&tcy);
}

```

```

/*****          bottom marking          *****/
bottom()
{
width = 13;
setlnwidth(&width);
setcolor(&liblue);
gcx=10.0,gcy=10.0;
movabs(&gcx,&gcy);
gcx=635.0;
lnabs(&gcx,&gcy);
var13();
settextclr(&white,&liblue);
tcx=450.0;tcy=7.0;
movtcurabs(&tcx,&tcy);
deltcur();
text("Hit any key to proceed");
}

```

```

bottom1()
{
var13();
settextclr(&white,&liblue);
tcx=150.0;tcy=7.0;

```

```

movtcurabs(&tcx,&tcy);
text("Hit");
tcx=198.0;
movtcurabs(&tcx,&tcy);
text("to see previous display.");
settextclr(&lired,&liblue);
tcx=182.0;
movtcurabs(&tcx,&tcy);
deltcur();
text("F");
}

```

```

/***** routine box function *****/
boxf()
{
box(&bx1,&by1,&bx2,&by2);
movabs(&gcx,&gcy);
flood(&blue);
}

```

```

/***** closing screen *****/
close()
{
h=7;w=5;p=0;m=0;
settext(&h,&w,&p,&m);
settextclr(&licyan,&black);
tcx=240.0;tcy=140.0;
movtcurabs(&tcx,&tcy);
deltcur();
text("END");
varl2();
}

```

```

/***** routine varing functions *****/
varl()
{
if(((3.14<theata)&&(theata<4.71))||((1.57<theata)&&(theata<
3.14)))
{
dx=dx*cos(theata)-dy*sin(theata);
dy=dx*sin(theata)+dy*cos(theata);
dy=-dy;
lnrel(&dx,&dy);
}
}

```

```

}
else
if(((0.0<theata)&&(theata<1.57))||((4.71<theata)&&(theata<6
.28)))
{
dx=dx*cos(theata)-dy*sin(theata);
dy=dx*sin(theata)+dy*cos(theata);
lnrel(&dx,&dy);
}
}

var2()
{
if (ht==0.0)
dx=dx;
else
dx=-dx;
lnrel(&dx,&dy);
}

var3()
{
trans_factor = 0.2 ;
if((tx<mx)&&(ty<my))
{
itx=0.0;ity=0.0;var5();
}
else if((tx<mx)&&(ty>my))
{
itx=0.0;ity=200.0;var5();
}
else if((tx>mx)&&(ty>my))
{
imx=0.0;imy=0.0;var4();
}
else if((tx>mx)&&(ty<my))
{
imx=0.0;imy=200.0;var4();
}
}

var4()
{
itx=imx+(tx-mx)*trans_factor*1.439;
ity=imy+(ty-my)*trans_factor;
}

var5()
{
imx=itx+(mx-tx)*trans_factor*1.439;
imy=ity+(my-ty)*trans_factor;
}

```

```

}

var6()
{
if(((3.14<theata)&&(theata<4.71))||((1.57<theata)&&(theata<
3.14)))
{
dx=dx*cos(theata)-dy*sin(theata);
dy=dx*sin(theata)+dy*cos(theata);
dy=-dy;
movrel(&dx,&dy);
}
else
if(((0.0<theata)&&(theata<1.57))||((4.71<theata)&&(theata<6
.28)))
{
dx=dx*cos(theata)-dy*sin(theata);
dy=dx*sin(theata)+dy*cos(theata);
movrel(&dx,&dy);
}
}

```

```

var7()
{
if (((tx<mx)&&(ty<my))&&(alpha<0.0))
{
imx=280.0;imy=40.0;
}
else if (((tx<mx)&&(ty<my))&&(alpha>0.0))
{
imx=120.0;imy=180.0;
}
else if (((tx<mx)&&(ty>my))&&(alpha<0.0))
{
imx=0.0;imy=10.0;
}
else if (((tx<mx)&&(ty>my))&&(alpha>0.0))
{
imx=280.0;imy=110.0;
}
else if (((tx>mx)&&(ty>my))&&(alpha<0.0))
{
imx=10.0;imy=160.0;
}
else if (((tx>mx)&&(ty>my))&&(alpha>0.0))
{
imx=140.0;imy=10.0;
}
else if (((tx>mx)&&(ty<my))&&(alpha<0.0))
{

```

```

    imx=200.0;imy=150.0;
  }
  else if (((tx>mx)&&(ty<my))&&(alpha>0.0))
  {
    imx=40.0;imy=40.0;
  }
}

var8()
{
  if (((tx<mx)&&(ty<my))&&(alpha<0.0))
  {
    itx=120.0;ity=130.0;
  }
  else if (((tx<mx)&&(ty<my))&&(alpha>0.0))
  {
    itx=130.0;ity=20.0;
  }
  else if (((tx<mx)&&(ty>my))&&(alpha<0.0))
  {
    itx=190.0;ity=170.0;
  }
  else if (((tx<mx)&&(ty>my))&&(alpha>0.0))
  {
    itx=120.0;ity=80.0;
  }
  else if (((tx>mx)&&(ty>my))&&(alpha<0.0))
  {
    itx=260.0;ity=70.0;
  }
  else if (((tx>mx)&&(ty>my))&&(alpha>0.0))
  {
    itx=220.0;ity=150.0;
  }
  else if (((tx>mx)&&(ty<my))&&(alpha<0.0))
  {
    itx=110.0;ity=30.0;
  }
  else if (((tx>mx)&&(ty<my))&&(alpha>0.0))
  {
    itx=110.0;ity=160.0;
  }
}

var9()
{
  dx=dx*1.439;
  lnrel(&dx,&dy);
  gcx=imx+80.0;gcy=imy+60.0;
  movabs(&gcx,&gcy);
}

```

```

var10()
{
  movtcurabs(&tcx,&tcy);
  text(s);
  deltcu();
}

```

```

var12()
{
  c=getch();
  setcolor(&black);
  clr();
}

```

```

var13()
{
  h=1;w=1;p=0;m=0;
  settex(&h,&w,&p,&m);
}

```

```

var14()
{
  width=1;
  setlnwidth(&width);
}

```

```

var15()
{
  if ( atan(sy/sx) < atan(vmtx/vmty) )
  {
    alpha1 = -alpha1 ;
    alpha2 = -alpha2 ;
  }
  else
  {
    alpha1 = alpha1 ;
    alpha2 = alpha2 ;
  }
}

```

```

var16()
{
  if (((vftx1<0.0)&&(vfty1>0.0))||((vftx1<0.0)&&(vfty1<0.0)))
    gammal=gammal+3.14;
}

```

```

else
    gamma1=gamma1+6.28;
if (gamma1>6.28)
    gamma1=gamma1-6.28;
else
    gamma1=gamma1;

if (((vftx2<0.0)&&(vfty2>0.0)) || ((vftx2<0.0)&&(vfty2<0.0)))
    gamma2=gamma2+3.14;
else
    gamma2=gamma2+6.28;
if (gamma2>6.28)
    gamma2=gamma2-6.28;
else
    gamma2=gamma2;
}

```



## APPENDIX C

### USER'S MANUAL

The following is the user's manual of ENDGAME.

USER'S MANUAL

AE 3251/AE 3705  
AIR DEFENSE LETHALITY

A STUDY  
of  
the ENDGAME  
in  
AIR DEFENSE

NAVAL POSTGRADUATE SCHOOL  
MONTEREY, CALIFORNIA

## I. INTRODUCTION

This homework problem is given to the student to present an opportunity for him/her to see how the vulnerability of an aircraft can be assessed in a missile-based air defense situation. The computer simulation program named ENDGAME will be used to assess the missile miss distance, the fragment spray zone and the best point for detonation.

## II. PROBLEM DEFINITION

You are going to conduct an endgame assessment of a generic attack aircraft on an attack mission to destroy a ship or ground vehicle. Each student will be given an ordnance package, target and missile velocity and flight path. This problem is purely for instructional purposes and is not based on any actual or planned combat situation. The aircraft vulnerability, missile characteristics, ordnance package, flight path parameter limits have been chosen only to provide guidelines for the class problem.

1. The initial positions of the target and the missile are restricted to within 100,000 ft and must be above ground. That is, the initial x and y positions of the target and the missile must be realistic. The error message which tells you " UNREAL! ASSUME THAT IT BE ABOVE GROUND AND REAL." comes if you type incorrect or unrealistic input data. Just input reasonable data.

2. The relative difference in distance between the initial x and y positions of the target and the missile must each lie within 1,000 ft. If this requirement is not met, no warning message will appear, but the results on the screen may be meaningless.

3. The target or aircraft speed must be greater than 0 and less than 2000 ft/sec ( or Mach 2 ). Otherwise, an error message tells you " UNREAL! ". Just type again the appropriate input.

4. Assume straight and level flight for the air target. The target heading, therefore, must be either  $0^{\circ}$  (left to right) or  $180^{\circ}$  (right to left).

5. The missile speed must be greater than the target speed. If not, the error message " UNREAL!" appears. So, use reasonable input data.

6. The input for the elevation angle of the missile,  $\theta$ , must also be reasonable. If the x and y positions, for example, of the target is greater than those of the missile, and if  $\theta$  is given as  $145^\circ$  or  $210^\circ$ , then the error message will warn you to use the appropriate elevation angle in order to allow the target to be approached.

### III. EXECUTING ENDGAME

You can execute ENDGAME on any IBM PC/AT personal computer or compatible with a monitor graphic resolution 640 x 350. If you have a different system on a personal computer, a few modifications to the program are required in order to run ENDGAME. Further details describing these modifications are given in this user's manual. The following instructions tell you how to run ENDGAME. Three different displays will come out from ENDGAME program.

1. Turn on the system power of your personal computer.
2. Insert the diskette which contains the execution file, the source file and the auxiliary files.
3. Run the executable file, endgame.exe, by simply typing "ENDGAME". Then, you will see the opening screen.
4. Just follow the instructions shown on the screen to do the simulation.
5. Do not hit any key to proceed until the message line comes on the bottom of the each screen. This means that the current display is being written to the specified file while the In-Use light on the diskette driver comes on.
6. You will be given the opportunity to see the previous screen by hitting "F" key when the best point for detonation screen is displayed.

#### IV. OUTPUT

Make sure that your printer is ready if you want to obtain the hard copy of the display. Just hit the "prt sc" key while holding the shift key, or hit the "print screen" key by itself on other special computer keyboards.

#### V. WAYS TO MODIFY THE PROGRAM

To modify the program in order to use the system that you have or you are going to use, you must make the following changes.

1. Among the external variables declared, the number of mode and page which are declared by integer should be varied by the systems that you have. That number depends upon how much memory is installed on the board and what kind of graphic device driver you have. See section 7 in Function Description of HALO manual (Version 2.26).

2. The graphics device driver must also be changed. See the HALO manual.

## LIST OF REFERENCES

1. Ball, Robert E., The Fundamentals of Aircraft Combat Survivability Analysis and Design, AIAA Education Series, New York, 1985.
2. Hearn, Donald and Baker, M.P., Computer Graphics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1986.
3. Lattice, Inc., Lattice C Compiler for MS-DOS (Version 3.00E), Lattice, Inc., Glen Ellyn, Illinois, 1985.

# INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3. Department Chairman, Code 67 Dept, of Aeronautics Naval Postgraduate School Monterey, California 93943-5000	1
4. Prof. Robert E. Ball, Code 67Bp Dept, of Aeronautics Naval Postgraduate School Monterey, California 93943-5000	2
5. Lee, Yeong Man 908 Ho, 206 Dong, Sam-ik Apt, Myoungil Dong, Kangdong Ku, Seoul, Republic of Korea	11
6. Air Force Central Library Sindaebang Dong, Kwanak Gu, Seoul, Republic of Korea	2
7. 3rd Department of Air Force College Sindaebang Dong, Kwanak Gu, Seoul, Republic of Korea	2
8. Library of Air Force Academy Chongwon Gun, Chung Cheong Bug Do, Republic of Korea	2
9. Commander (Attn: John Morrow, Code 338) Naval Weapons Center China Lake, California 93555-6001	1
10. Commander (Attn: Dale Atkinson, AIR-5164) Naval Air Systems Command Washington D.C. 20361-5160	1

- |     |  |   |
|-----|--|---|
| 11. | John Vice                                    | 1 |
|     | AFWAL/FIES/SURVIAC                           |   |
|     | Wright-Patterson AFB, Ohio 45433-6553        |   |
| 12. | Charles Alston                               | 1 |
|     | AMXSY-AD                                     |   |
|     | Aberdeen Proving Ground, Maryland 21005-5071 |   |



END  
FILMED  
FEB. 1988  
DTIC